CLAIMS

1. An interface adapter for a packet network, comprising:

a memory interface, for coupling to a memory;

a first plurality of execution engines, coupled to the host interface so as to read from the memory work items corresponding to messages to be sent over the network, and to generate gather entries defining packets to be transmitted over the network responsive to the work items; and

a scheduling processor, coupled to assign the work items to the execution engines for generation of the gather entries;

a second plurality of gather engines, which are adapted to generate the packets responsive to the gather entries; and

switching circuitry, coupling the execution engines to the gather engines so as to submit the gather entries to the gather engines for generation of the packets responsive thereto.

2. An adapter according to claim 1, wherein each of the work items belongs to a respective transport service instance among multiple transport service instances served by the adapter, and wherein the scheduling processor is adapted to assign the work items to the execution engines by selecting the respective transport service instances for service.

3. An adapter according to claim 2, wherein the transport service instances comprise queue pairs, and wherein the work items comprise work queue elements.

4. An adapter according to claim 2, wherein for each of the transport service instances, the respective work items are maintained in a list in the memory, and wherein the execution engines are configured so as to generate in succession the gather entries corresponding to the work items in the list maintained for the transport service instance selected by the scheduling processor.

5. An adapter according to claim 2, wherein the scheduling processor is adapted to maintain a third plurality of scheduling queues in which are entered the transport service instances to which the work items belong, the scheduling queues having respective heads and tails, and to select the instances from the heads of the queues for assignment to the execution engines.

6. An adapter according to claim 5, wherein the scheduling queues are associated with respective classes of service provided by the adapter, and wherein the transport service instances are assigned to the scheduling queues according to the classes of services to which the corresponding transport services belong.

7. An adapter according to claim 6, wherein the scheduling processor is adapted to determine the scheduling queues from which the transport service instances are to be assigned to the execution engines in accordance with a scheduling policy relating to the respective classes of service of the scheduling queues.

8. An adapter according to claim 7, wherein the switching circuitry is coupled to arbitrate among the execution engines so as to submit the gather entries to the gather engines in an order responsive to the classes of service.

9. An adapter according to claim 5, wherein the scheduling processor is further adapted to enter the transport service instances at the tails of the scheduling queues to which they are assigned when work items belonging to the transport service instances are written to the memory.

10. An adapter according to claim 5, wherein context information regarding each of the transport service instances is recorded in the memory, and wherein the scheduling processor is adapted to maintain the scheduling queues by directing pointers in the context information of each of the instances, except the instances at the tails of the scheduling queues, to point to the context information of succeeding instances in the queues.

11. An adapter according to claim 2, wherein the transport service instances are assigned to respective classes of service provided by the adapter, and wherein the scheduling processor is adapted to select the transport service instances for service by the execution engines responsive to the assigned classes of service.

12. An adapter according to claim 11, wherein the switching circuitry is coupled to arbitrate among the execution engines so as to submit the gather entries to the gather engines in an order responsive to the classes of service.

13. An adapter according to claim 1, wherein the work items belong to different transport service instances, which are associated with respective classes of service, and wherein the scheduling processor is adapted to select

the work items to be assigned to the execution engines responsive to the classes of service.

14. An adapter according to claim 13, wherein the switching circuitry is coupled to arbitrate among the execution engines so as to submit the gather entries to the gather engines in an order responsive to the classes of service.

15. An adapter according to claim 13, and comprising an execution access arbiter, coupled between the execution engines and the memory interface so as to control an order of access to the memory by the execution engines in reading the work items, responsive to the classes of service.

16. An adapter according to claim 1, wherein the execution engines are adapted to generate multiple gather entries corresponding to a single one of the work items, each of the gather entries defining no more than a single packet to be generated by one of the gather engines.

17. An adapter according to claim 1, wherein the work items comprise descriptors indicating data to be read from the memory for inclusion in the packets, and wherein the gather engines are coupled to read the data from the memory by direct memory access (DMA).

18. An adapter according to claim 17, wherein the messages comprise remote direct memory access (RDMA) requests, and wherein the work items are written to the memory by a host processor submitting the requests.

19. An adapter according to claim 17, wherein the messages comprise remote direct memory access (RDMA) responses, and wherein the work items are written to the

memory responsive to RDMA request packets received by the adapter via the network.

20. An adapter according to claim 17, and comprising a data access arbiter, coupled between the gather engines and the memory interface so as to control an order of access to the memory by the gather engines for reading the data.

21. An adapter according to claim 20, wherein the work items belong to different transport service instances, which are associated with respective classes of service, and wherein the data access arbiter is programmable so as to give priority in the order of access to one or more of the gather engines responsive to the classes of service.

22. An adapter according to claim 1, wherein the work items belong to different transport service instances, and wherein the scheduling processor comprises a microprocessor, which is programmed by software code to select the transport service instances for assignment to the execution engines, and wherein the switching circuitry comprises:

one or more registers, containing respective weight factors associated with the execution engines; and

one or more hardware logic circuits, coupled to arbitrate among the execution engines to submit the gather entries to each of the gather engines responsive to the weight factors.

23. An adapter according to claim 22, wherein the memory interface, execution engines, scheduling processor, gather engines and switching circuitry are comprised in a single integrated circuit chip.

24. An adapter according to claim 22, wherein the one or more hardware logic circuits comprise a respective arbiter circuit connected to each one of the second plurality of gather engines, and wherein the arbiter circuit is coupled to arbitrate among the execution engines whose gather entries are to be submitted to the one of the gather engines connected thereto.

25. An adapter according to claim 24, wherein the scheduling processor is adapted to determine which of the execution engines are to submit the gather entries to each of the gather engines, and to assign the execution engines to submit the gather entries accordingly, and wherein the arbiter circuit connected to each of the gather engines is coupled to arbitrate among the execution engines assigned thereto.

26. An adapter according to claim 1, wherein the first plurality is greater than the second plurality.

27. An adapter according to claim 1, wherein the packets generated by the gather engines are transmitted over the network at a given transmission speed of the network, and wherein the first plurality and the second plurality are chosen so as to be capable of generating the packets at a speed greater than the given transmission speed.

28. An interface adapter for a packet network, comprising:

a memory interface, for coupling to a memory, to which work items belonging to respective transport service instances are written, wherein the transport service instances are assigned to respective classes of service, and the work items indicate messages to be sent over the network using the transport service instances;

a plurality of execution engines, coupled to the memory interface so as to read the work items from the memory, and to generate gather entries defining packets to be transmitted over the network responsive to the work items;

a scheduling processor, adapted to maintain a multiplicity of scheduling queues corresponding to the classes of service, the scheduling queues having respective heads and tails, and further adapted to enter the transport service instances to which the work items belong in the scheduling queues according to the classes of service of the instances, and to select the instances from the heads of the queues for assignment to the execution engines for generation of the gather entries; and

a send data unit, which is adapted to generate the packets responsive to the gather entries.

29. An adapter according to claim 28, wherein the transport service instances comprise queue pairs, and wherein the work items comprise work queue elements.

30. An adapter according to claim 28, wherein for each of the transport service instances, the respective work items are maintained in a list in the memory, and wherein the execution engines are configured so as to generate in succession the gather entries corresponding to the work items in the list maintained for the transport service instance selected by the scheduling processor.

31. An adapter according to claim 28, wherein the scheduling processor is adapted to determine the scheduling queues from which the transport service instances are to be assigned to the execution engines in

accordance with a scheduling policy relating to the respective classes of service of the scheduling queues.

32. An adapter according to claim 28, wherein the scheduling processor is further adapted to enter the transport service instances at the tails of the scheduling queues to which they are assigned when work items belonging to the transport service instances are written to the memory.

33. An adapter according to claim 32, wherein the scheduling processor is adapted to enter the transport service instance in the scheduling queues responsive to a host processor having written to a doorbell address of the adapter.

34. An adapter according to claim 28, wherein context information regarding each of the transport service instances is recorded in the memory, and wherein the scheduling processor is adapted to maintain the scheduling queues by directing pointers in the context information of each of the instances, except the instances at the tails of the scheduling queues, to point to the context information of succeeding instances in the queues.

35. An adapter according to claim 28, wherein the messages comprise remote direct memory access (RDMA) requests, and wherein the work items are written to the memory by a host processor submitting the requests.

36. An adapter according to claim 28, wherein the messages comprise remote direct memory access (RDMA) responses, and wherein the work items are written to the

memory responsive to RDMA request packets received by the adapter via the network.

37. An adapter according to claim 28, wherein at least some of the messages comprise data to be read from the memory and sent to a recipient via the network, and wherein the work items indicate the data to be sent.

38. An adapter according to claim 28, and comprising an execution access arbiter, coupled between the execution engines and the memory interface so as to control an order of access to the memory by the execution engines in reading the work items, responsive to the classes of service.

39. An adapter according to claim 28, wherein each of the execution engines comprises:

a buffer for holding the work items to be processed, the buffer having a programmable watermark level;

an execute machine, coupled to read the work items from the buffer and to generate the gather entries corresponding thereto; and

a fetch machine, coupled to fetch the work items from the memory to the buffer responsive to a volume of the work items in the buffer having fallen below the watermark level.

40. An adapter according to claim 39, wherein the watermark level is programmable responsive to the classes of service of the transport service instances assigned to each of the execution engines for generation of the corresponding gather entries.

41. An interface adapter for a packet network, comprising:

a memory interface, for coupling to a memory, to which are written work items belonging to different transport service instances and corresponding to messages to be sent over the network;

a plurality of execution engines, which are adapted to process the work items so as to generate gather entries defining packets to be transmitted over the network, each such execution engine comprising:

a buffer for holding the work items to be processed by the execution engine, the buffer having a programmable watermark level;

an execute machine, coupled to read the work items from the buffer and to generate the gather entries responsive thereto; and

a fetch machine, coupled to fetch the work items from the memory to the buffer responsive to a volume of the work items in the buffer having fallen below the watermark level;

a scheduling processor, coupled to assign the transport service instances to the execution engines for generation of the gather entries based on the work items belonging to the transport service instances; and

one or more gather engines, which are adapted to generate the packets responsive to the gather entries.

42. An adapter according to claim 41, wherein each of the transport service instances is associated with a respective class of service, and wherein the watermark level in each of the execution engines is programmable responsive to the class of service of the transport service instances assigned to the execution engine.

43. An adapter according to claim 42, and comprising an execution access arbiter, coupled between the fetch machine in each of the execution engines and the memory interface so as to control an order of access to the memory by the fetch machines in fetching the work items, responsive to the classes of service.

44. A method for communicating over a packet network, comprising:

receiving work items corresponding to messages to be sent over the network;

assigning each of the work items to one of a first plurality of execution engines;

generating gather entries using the assigned execution engines, the gather entries defining packets to be transmitted over the network responsive to the work items;

arbitrating among the first plurality of the execution engines so as to distribute the gather entries among a second plurality of gather engines; and

generating the packets using the gather engines responsive to the gather entries.

45. A method according to claim 44, wherein each of the work items belongs to a respective transport service instance among multiple transport service instances on the network, and wherein assigning the work items comprises selecting the respective transport service instances for service.

46. A method according to claim 45, wherein the transport service instances comprise queue pairs, and wherein the work items comprise work queue elements.

47. A method according to claim 45, wherein generating the gather entries comprises reading in succession the work items in a list of the work items maintained in a memory for each of the transport service instances, and generating a sequence of the gather entries corresponding to the work items in the list.

48. A method according to claim 45, wherein assigning each of the work items comprises:

maintaining a third plurality of scheduling queues having respective heads and tails;

entering the transport service instances to which the work items belong in the queues; and

selecting the instances from the heads of the queues for assignment to the execution engines.

49. A method according to claim 48, wherein maintaining the third plurality of the scheduling queues comprises associating the scheduling queues with respective classes of communication service, and assigning the transport service instances to the scheduling queues according to the classes of services to which the corresponding transport services belong.

50. A method according to claim 49, wherein selecting the instances for assignment to the execution engines comprises assigning the execution engines in accordance with a scheduling policy relating to the respective classes of service of the scheduling queues.

51. A method according to claim 50, wherein arbitrating among the first plurality of the execution engines comprises arbitrating among the execution engines so as to submit the gather entries to the gather engines in an order responsive to the classes of service.

52. A method according to claim 48, wherein receiving the work items comprises entering the transport service instances at the tails of the scheduling queues to which they are assigned when work items belonging to the transport service instances are received.

53. A method according to claim 48, wherein context information regarding each of the transport service instances is recorded in a memory, and wherein entering the transport service instances in the queues comprises directing pointers in the context information of each of the instances, except the instances at the tails of the scheduling queues, to point to the context information of succeeding instances in the queues.

54. A method according to claim 45, wherein the transport service instances are assigned to respective classes of service, and wherein selecting the respective transport service instances comprises choosing the instances responsive to the assigned classes of service.

55. A method according to claim 54, wherein arbitrating among the first plurality of the execution engines comprises submitting the gather entries to the gather engines in an order responsive to the classes of service.

56. A method according to claim 44, wherein the work items belong to different transport service instances, which are associated with respective classes of service, and wherein assigning each of the work items comprises assigning the transport service instances to the execution engines responsive to the classes of service.

57. A method according to claim 56, wherein arbitrating among the first plurality of the execution engines

comprises submitting the gather entries to the gather engines in an order responsive to the classes of service.

58. A method according to claim 56, wherein generating the gather entries comprises reading the work items from the memory using the execution engines, and comprising controlling an order of access to the memory by the execution engines in reading the work items, responsive to the classes of service.

59. A method according to claim 44, wherein generating the gather entries comprises generating multiple gather entries corresponding to a single one of the work items, such that each of the gather entries defines no more than a memory read operation to be performed by one of the gather engines.

60. A method according to claim 44, wherein the work items comprise descriptors indicating data to be read from a memory for inclusion in the packets, and wherein generating the packets comprises reading the data from the memory using the gather engines by direct memory access (DMA).

61. A method according to claim 60, wherein receiving the work items comprises receiving remote direct memory access (RDMA) requests submitted by a host processor.

62. A method according to claim 60, wherein receiving the work items comprises receiving indications of remote direct memory access (RDMA) responses to be made responsive to RDMA request packets received via the network.

63. A method according to claim 60, wherein in at least some of the messages, the data read from the memory are

sent to a recipient via the network, and wherein receiving the work items comprises receiving descriptors indicating the data to be sent.

64. A method according to claim 60, wherein the work items belong to different transport service instances, which are associated with respective classes of service, and comprising controlling an order of access to the memory by the gather engines for reading the data so as to give priority in the order of access to one or more of the gather engines responsive to the classes of service.

65. A method according to claim 44, wherein the first plurality is greater than the second plurality.

66. A method according to claim 44, wherein the packets generated by the gather engines are transmitted over the network at a given transmission speed of the network, and wherein the first plurality and the second plurality are chosen so as to be capable of generating the packets at a speed greater than the given transmission speed.

67. A method for communicating over a packet network, comprising:

receiving work items belonging to respective transport service instances, wherein the transport service instances are assigned to respective classes of service, and the work items indicate messages to be sent over the network using the transport service instances;

creating a multiplicity of scheduling queues corresponding to the classes of service, the scheduling queues having respective heads and tails;

entering the transport service instances to which the work items belong in the scheduling queues according to the classes of service of the instances;

selecting the instances at the heads of the queues to be assigned for service by a plurality of execution engines;

generating gather entries using the assigned execution engines, the gather entries defining packets to be transmitted over the network responsive to the work items; and

generating the packets responsive to the gather entries.

68. A method according to claim 67, wherein the transport service instances comprise queue pairs, and wherein the work items comprise work queue elements.

69. A method according to claim 67, wherein generating the gather entries comprises reading in succession the work items in a list of the work items maintained in a memory for each of the transport service instances, and generating a sequence of the gather entries corresponding to the work items in the list maintained for the selected instances.

70. A method according to claim 67, wherein selecting the instances comprises determining the scheduling queues from which the transport service instances are to be selected for service by the execution engines in accordance with a scheduling policy relating to the respective classes of service of the scheduling queues.

71. A method according to claim 67, wherein receiving the work items comprises entering the transport service instances at the tails of the scheduling queues to which they are assigned when work items belonging to the transport service instances are received.

72. A method according to claim 67, wherein context information regarding each of the transport service instances is recorded in the memory, and wherein entering the transport service instances in the scheduling queues comprises directing pointers in the context information of each of the instances, except the instances at the tails of the scheduling queues, to point to the context information of succeeding instances in the queues.

73. A method according to claim 67, wherein receiving the work items comprises receiving remote direct memory access (RDMA) requests submitted by a host processor.

74. A method according to claim 67, wherein receiving the work items comprises receiving indications of remote direct memory access (RDMA) responses to be made responsive to RDMA request packets received via the network.

75. A method according to claim 67, wherein at least some of the messages comprise data to be read from the memory and sent to a recipient via the network, and wherein receiving the work items comprises receiving descriptors indicating the data to be sent.

76. A method according to claim 67, wherein generating the gather entries comprises reading the work items from the memory using the execution engines, and comprising controlling an order of access to the memory by the execution engines in reading the work items, responsive to the classes of service.

77. A method according to claim 67, wherein generating the gather entries comprises:

providing each of the execution engines with a buffer for holding the work items to be used in generating the gather entries, the buffer having a watermark level;

setting the watermark level responsive to the classes of service of the transport service instances assigned to each of the execution engines;

reading the work items out of the buffer so as to generate the gather entries corresponding thereto; and

fetching the work items into the buffers responsive to a volume of the work items in the buffer having fallen below the watermark level.

78. A method for communicating over a packet network, comprising:

receiving work items corresponding to messages to be sent over the network;

assigning each of the work items to one of a plurality of execution engines, each of the execution engines having a buffer for holding the work items to be used in generating the gather entries, the buffer having a programmable watermark level;

reading the work items out of the buffer so as to generate gather entries defining packets to be transmitted over the network responsive to the work items;

fetching the work items into the buffer responsive to a volume of the work items in the buffer having fallen below the watermark level; and

generating the packets responsive to the gather entries.

79. A method according to claim 78, wherein each of the work items belongs to different transport service instances, which is associated with a respective class of service, and comprising setting the watermark level in each of the execution engines is responsive to the class of service of the work items assigned to the execution engine.

80. A method according to claim 78, wherein fetching the work items comprises arbitrating among the execution engines so as to control an order of access to the memory by the execution engines in fetching the work items, responsive to the classes of service.